10/540219

PCT/FR03/03658

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

REC'D 23 FEB 2004
WIPO PCT

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 16 DEC. 2003

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

DOCUMENT DE PRIORITÉ

PRÉSENTÉ OU TRANSMIS CONFORMÉMENT À LA RÈGLE 17.1.a) OU b)

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (0)1 53 04 53 04
Télécopie : 33 (0)1 53 04 45 23
www.inpi.fr

ETABLISSEMENT PUBLIC NATIONAL CREE PAR LA LOI N° 51-444 DU 19 AVRIL 1951

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 33 (1) 53 04 53 04 Télécopie : 33 (1) 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11354*02

REQUÊTE EN DÉLIVRANCE
page 1/2

BR1

Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 W / 010502

Réservé à l'INPI

REMISE DES PIÈCES
DATE
LIEU 20 DEC 2002
75 INPI PARIS
N° D'ENREGISTREMENT
NATIONAL ATTRIBUÉ PAR L'INPI 0216376
DATE DE DÉPÔT ATTRIBUÉE
PAR L'INPI 20 DEC. 2002

Vos références pour ce dossier *(facultatif)*
BIF114681/FR

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

CABINET BONNET-THIRION
12, avenue de la Grande Armée,
75017 PARIS

2 NATURE DE LA DEMANDE — **Cochez l'une des 4 cases suivantes**

Demande de brevet	☒	
Demande de certificat d'utilité	☐	
Demande divisionnaire	☐	
Demande de brevet initiale	N°	Date
ou demande de certificat d'utilité initiale	N°	Date
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N°	Date

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Procédé et dispositif de sécurisation de l'exécution d'un programme informatique.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE

Pays ou organisation
Date N°
Pays ou organisation
Date N°
Pays ou organisation
Date N°

☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**

5 DEMANDEUR (Cochez l'une des 2 cases) ☒ **Personne morale** ☐ **Personne physique**

Nom ou dénomination sociale		OBERTHUR CARD SYSTEMS SA
Prénoms		
Forme juridique		Société anonyme
N° SIREN		
Code APE-NAF		
Domicile ou siège	Rue	102, Boulevard Malesherbes,
	Code postal et ville	7 5 0 1 7 PARIS
	Pays	FRANCE
Nationalité		FRANCAISE
N° de téléphone *(facultatif)*		N° de télécopie *(facultatif)*
Adresse électronique *(facultatif)*		

☐ **S'il y a plus d'un demandeur, cochez la case et utilisez l'imprimé «Suite»**

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE
page 2/2

BR2

Réservé à l'INPI

REMISE DES PIÈCES
DATE 20 DEC 2002
LIEU 75 INPI PARIS
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0216376

BIF114681/FR

DB 540 W / 200502

6 MANDATAIRE *(s'il y a lieu)*		
Nom		
Prénom		
Cabinet ou Société		CABINET BONNET-THIRION
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	12, Avenue de la Grande Armée,
	Code postal et ville	7 5 0 1 7 PARIS
	Pays	
N° de téléphone *(facultatif)*		01 53 81 17 00
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		**Les inventeurs sont nécessairement des personnes physiques**
Les demandeurs et les inventeurs sont les mêmes personnes		☐ Oui ☒ Non : **Dans ce cas remplir le formulaire de Désignation d'inventeur(s)**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☒ ☐
Paiement échelonné de la redevance *(redevance inpi)*		**Uniquement pour les personnes physiques effectuant elles-mêmes leur propre dépôt** ☐ Oui ☐ Non
9 RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Obtenue antérieurement à ce dépôt pour cette invention *(joindre une copie de la décision d'admission à l'assistance gratuite ou indiquer sa référence)* : AG
10 SÉQUENCES DE NUCLEOTIDES ET/OU D'ACIDES AMINÉS		☐ Cochez la case si la description contient une liste de séquences
Le support électronique de données est joint		☐
La déclaration de conformité de la liste de séquences sur support papier avec le support électronique de données est jointe		☐
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

11 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)

Bruno QUANTIN N°92.1206
CABINET BONNET-THIRION

VISA DE LA PRÉFECTURE OU DE L'INPI

C. CONTE

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire.
Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

La présente invention se rapporte à un procédé et à un dispositif de sécurisation de l'exécution d'un programme informatique.

Elle se rapporte également à un dispositif de traitement d'un programme informatique pouvant être sécurisé avec un procédé ou un dispositif conforme à la présente invention.

Le procédé et le dispositif de sécurisation conformes à l'invention peuvent notamment, mais de façon non limitative être utilisés pour sécuriser l'exécution d'un programme sur une carte à microcircuit.

Dans la suite de ce document on entendra par « sécurisation » d'un programme informatique :

- la détection d'attaques mal intentionnées visant à modifier le comportement normal d'un programme informatique ; mais aussi
- la fiabilisation du déroulement d'un programme informatique notamment celle d'un programme s'exécutant dans un environnement très perturbé, (comme un satellite) ou celle d'un programme informatique à forte exigence de fiabilité comme par exemple un programme de contrôle d'un implant cardiaque.

La présente invention permet en particulier de détecter une attaque destinée à modifier le déroulement de l'exécution d'un programme informatique s'exécutant dans une carte à microcircuit.

Elle permet en particulier de détecter des attaques par perturbation du fonctionnement de la carte à microcircuit, souvent appelées DFA (de l'anglais "Differential Fault Analysis").

Ces attaques visent à modifier illicitement le contenu d'un registre, d'une mémoire ou d'un bus, ou à obliger un processeur à ne pas exécuter certaines instructions du programme informatique. Dans ce cas, le programme informatique attaqué peut se dérouler d'une façon très différente de celle qui avait été prévue au moment de sa conception.

Elles peuvent, entre autres et de façon connue, être effectuées :

- en générant un pic de tension à l'une des bornes d'alimentation du processeur ;

- en élevant brusquement sa température;

- en changeant rapidement sa fréquence d'horloge ou sa tension d'alimentation ;

- en appliquant un flash de lumière ou un rayon laser sur une partie du silicium qui le compose.

Selon l'état actuel de la technique, l'homme du métier dispose de différents moyens pour lutter contre les attaques DFA.

Il existe en particulier dans la plupart des composants des cartes à microcircuit, des capteurs qui permettent de détecter de telles attaques. Leur efficacité est néanmoins restreinte car il est en pratique impossible de mettre des capteurs sur toute la surface de ce composant.

Par ailleurs ces capteurs étant également composés de silicium, il est possible de les perturber ou de modifier les informations qu'ils transmettent.

On connaît d'autre part, par le document WO 00/65442 un mécanisme de protection du cœur d'un processeur contre des manipulations externes.

Ce document décrit un procédé de sécurisation d'un programme informatique consistant, au cours de l'exécution d'une première instruction, à calculer une première valeur à partir du contenu des registres du processeur, puis une deuxième valeur selon le même principe juste avant l'exécution de l'instruction immédiatement suivante.

Par comparaison de ces deux valeurs, ce procédé permet de détecter une attaque éventuelle qui aurait été commise dans l'intervalle de temps s'écoulant entre la fin de la première instruction et le début de l'exécution suivante, soit pendant un moment d'inactivité du processeur.

Ce procédé de sécurisation présente néanmoins deux inconvénients :

Tout d'abord, il ne permet pas la détection d'une attaque qui se produirait au cours de l'exécution d'une instruction ou pendant le chargement (FETCH) du code d'une instruction dans un registre de calcul du processeur.

En effet, si les registres du processeur restent inchangés entre deux instructions consécutives, ce procédé ne détectera pas d'anomalie, même si ces registres contiennent une valeur illicite, due par exemple à une attaque au cours de la première instruction, juste avant le calcul de la première valeur.

D'autre part, le résultat de calcul de la première valeur étant a priori inconnu, ce procédé impose que la détection d'une attaque potentielle consécutive à une instruction soit effectuée avant même l'exécution de l'instruction suivante, car l'exécution de cette instruction suivante modifierait nécessairement la valeur des registres du processeur et donc la valeur.

Ainsi, seule une attaque se produisant entre deux instructions peut être détectée, mais pas un déroulement inattendu du programme, comme par exemple un saut illicite.

En particulier, il ne permet pas de mémoriser l'historique des instructions ayant été exécutées avant une instruction sensible, c'est-à-dire une instruction dont l'attaque met en péril la sécurité, comme par exemple des instructions de comparaison de données secrètes lors de la vérification d'un code secret (PIN). Cet inconvénient est particulièrement gênant car il empêche de se prémunir contre une attaque par perturbation du processeur, par exemple de type DFA, qui forcerait le processeur à exécuter cette instruction sensible sans exécuter les instructions préalables prévues.

Plus précisément, et selon un premier aspect, la présente invention concerne un procédé de sécurisation de l'exécution d'un programme informatique comportant un ensemble d'au moins une instruction.

Dans la suite de ce document on conviendra d'appeler "signature", toute information représentative d'opérations prédéterminées effectuées par un processeur lors de l'exécution d'un ensemble d'instructions informatiques.

Le procédé de sécurisation selon l'invention est caractérisé en ce qu'il comporte :

-une première étape de calcul et de mémorisation, préalable à l'exécution du programme informatique, d'une première signature représentative de l'exécution attendue de l'ensemble d'instructions;

-une deuxième étape de calcul et de mémorisation, au cours de l'exécution de l'ensemble d'instructions, d'une deuxième signature représentative de l'exécution de l'ensemble d'instructions ; et

-une étape de détection d'une anomalie d'exécution de l'ensemble d'instructions à partir des première et deuxième signatures.

Préférentiellement, la première étape de calcul et de mémorisation a lieu au cours de la génération des instructions du programme informatique.

Par exemple, cette première étape de calcul et de mémorisation peut être effectuée à partir du code assembleur des instructions du programme informatique ou à partir du code compilé ou du code exécutable de ce programme.

Ainsi, conformément à ce premier aspect de l'invention, la première signature est calculée et mémorisée au moment de la génération du programme informatique. La valeur de cette première signature, représentative de l'exécution normale du programme est donc connue.

Lors de l'exécution du programme, on calcule une deuxième signature représentative de l'exécution réelle des instructions du programme. Cette deuxième signature est également mémorisée.

Il est donc possible, grâce à l'invention, de détecter une attaque se produisant au cours de l'exécution d'une instruction du programme ou pendant le chargement (FETCH) du code de cette instruction dans un registre de calcul du processeur.

En effet, si tel était le cas, la deuxième signature serait différente de la première.

Dans une première variante préférée de réalisation du procédé de sécurisation selon l'invention, la mémorisation de la deuxième signature est conservée pendant l'exécution d'au moins une deuxième instruction consécutive à l'ensemble d'instructions.

Ainsi, la détection de l'anomalie d'exécution peut être réalisée ultérieurement, y compris après l'exécution de plusieurs instructions consécutives à l'ensemble d'instructions.

Ceci permet de façon très avantageuse de mémoriser l'historique des instructions ayant été exécutées avant une instruction sensible, et donc de détecter les attaques visant à obliger le processeur à éviter l'exécution d'un ensemble d'instructions données.

Ce mode de réalisation permet aussi avantageusement d'effectuer la détection d'une attaque à un moment quelconque de l'exécution du programme ce qui rend très difficile une attaque sur cette détection proprement dite.

Dans un premier mode préféré de réalisation, la première et la deuxième signatures sont obtenues à partir du nombre d'instructions de l'ensemble d'instructions.

Selon ce premier mode préféré de réalisation, le procédé de sécurisation permet de détecter un saut inattendu dans l'exécution du programme informatique.

Selon un premier exemple de ce premier mode préféré, la première signature est le nombre d'instructions de l'ensemble d'instructions, la deuxième signature est le nombre d'instructions de l'ensemble d'instructions ayant été exécutées à un moment donné, l'étape de détection détectant une anomalie d'exécution lorsque à l'issue de l'exécution de l'ensemble d'instructions la première et la deuxième signature diffèrent.

En variante de ce premier exemple de ce premier mode préféré, l'étape de détection détecte une anomalie uniquement lorsque la deuxième signature est strictement inférieure à la première signature. Ceci permet avantageusement de ne détecter une anomalie que dans le cas où un nombre minimum d'instructions n'a pas été exécuté.

Dans un deuxième exemple de ce premier mode préféré de réalisation, la première et la deuxième signatures sont constituées par le nombre d'instructions de l'ensemble d'instructions et des arguments de ces instructions.

Selon un troisième exemple de ce premier mode préféré de réalisation, la deuxième signature est obtenue à partir du nombre d'instructions de l'ensemble d'instructions n'ayant pas été exécutées.

Cette deuxième signature peut être obtenue en soustrayant à la première signature mémorisée au cours de la première étape de calcul et de mémorisation, le nombre d'instructions ayant déjà été exécutées par le processeur. Dans cette variante, l'étape de détection détecte une anomalie lorsque à l'issue de l'exécution de l'ensemble d'instructions, la deuxième signature est non nulle.

Selon un quatrième exemple de ce premier mode préféré de réalisation, la première signature est égale au nombre d'instructions de l'ensemble d'instructions augmenté du nombre d'arguments de ces instructions, la deuxième signature étant égale au nombre d'instructions déjà exécutées augmenté du nombre d'arguments de ces instructions.

Avantageusement, dans ces troisième et quatrième exemples du premier mode préféré de réalisation on déclenche une instruction du programme informatique lorsque la deuxième signature est inférieure à un seuil prédéterminé.

On peut ainsi détecter une exécution inattendue du programme informatique même si l'étape de détection proprement dite est rendue inefficace du fait d'une attaque visant cette étape elle-même.

Dans un mode préféré de ces troisième et quatrième exemples du premier mode de réalisation, la première et la deuxième signatures sont conservées en mémoire pendant l'exécution du programme, dans le même registre.

Ce mode de réalisation est avantageux car il permet d'économiser un registre de mémoire.

Dans un deuxième mode préféré de réalisation :

-la première signature est obtenue à partir du code d'une instruction critique de l'ensemble d'instructions ;

-la deuxième signature est obtenue à partir du code de l'instruction critique, ce code étant mémorisé simultanément ou consécutivement à l'exécution de cette instruction critique ; et

- l'étape de détection détecte une anomalie d'exécution lorsque, à l'issue de l'exécution de l'ensemble d'instructions, les première et deuxième signatures sont différentes.

Préférentiellement, ce code d'instruction est, en pratique, le code machine de l'instruction reconnu par le processeur.

En variante ces signatures peuvent par exemple être constituées par le code d'instruction et ses arguments.

Ce deuxième mode préféré de réalisation, permet de vérifier que l'instruction critique a effectivement été exécutée lors de l'exécution du programme informatique.

Selon un troisième mode préféré de réalisation :

-la première signature est obtenue à partir de l'adresse d'une d'instruction critique de l'ensemble d'instructions, cette adresse étant obtenue pendant ou après la génération du code exécutable de l'ensemble d'instructions;

-la deuxième signature est obtenue à partir de l'adresse de cette instruction critique, cette adresse étant mémorisée simultanément ou consécutivement à l'exécution de cette instruction critique ; et

-l'étape de détection détecte une anomalie d'exécution lorsque, à l'issue de l'exécution de l'ensemble d'instructions, les première et deuxième signatures sont différentes.

Dans ce troisième mode préféré de réalisation, la première et la deuxième signatures peuvent être par exemple constituées par l'adresse de l'instruction proprement dite, cette adresse pouvant être complétée par l'adresse d'un argument de cette instruction.

Ce troisième mode préféré de réalisation permet également de vérifier que l'instruction critique a été exécutée.

Dans ce troisième mode de réalisation, l'instruction critique peut être une instruction de saut, comme par exemple les instructions JMP, JNZ, CJNE, JZ en langage assembleur .

Il est ainsi possible de vérifier que des instructions en cours d'exécution ne sont pas exécutées à la suite d'une attaque sur une instruction de saut.

Selon un quatrième mode de réalisation préféré :

-les première et deuxième signatures sont des codes détecteurs d'erreur calculés à partir du code ou de l'adresse d'au moins une instruction de l'ensemble d'instructions ; et

-l'étape de détection détecte une anomalie d'exécution lorsque, à l'issue de l'exécution de l'ensemble d'instructions, les première et deuxième signatures sont différentes.

Ce mode préféré de réalisation est particulièrement avantageux lorsque l'on veut vérifier que plusieurs instructions de l'ensemble d'instructions ont été exécutées, ces instructions pouvant notamment ne pas être consécutives dans cet ensemble.

Dans un autre mode de réalisation les codes détecteurs d'erreur sont obtenus par combinaison logique des codes d'instructions, par exemple un OU exclusif logique (XOR).

De façon préférée, les codes détecteurs d'erreur précités sont des codes de redondance cycliques (CRC), ce mode de réalisation étant à la fois particulièrement facile à mettre en œuvre, et d'exécution rapide.

Dans un cinquième mode préféré de réalisation, la première et la deuxième signatures sont obtenues, respectivement au cours de la génération et de l'exécution desdites instructions, à partir d'au moins deux éléments obtenus à partir :

-du nombre d'instructions de l'ensemble d'instructions; ou

-du code d'au moins une instruction de cet ensemble d'instructions; ou

-de l'adresse d'au moins une instruction de cet ensemble d'instructions; ou

- d'un code détecteur d'erreur calculé à partir du code ou d'une adresse d'au moins une instruction critique de cet ensemble d'instructions, cette adresse étant obtenue pendant ou après la génération du code exécutable de l'ensemble d'instructions.

Dans ce mode préféré de réalisation, l'étape de détection détecte une anomalie d'exécution lorsque, à l'issue de l'exécution de l'ensemble d'instructions, la première et la deuxième signatures sont différentes.

Les première et deuxième signatures de ce cinquième mode de réalisation combinent ainsi les avantages des signatures décrites brièvement ci dessus en référence avec quatre premiers modes de réalisation.

Dans un autre mode préféré de réalisation, le procédé de sécurisation comporte en outre une étape de destruction d'au moins une partie du système sur lequel s'exécute le programme informatique, cette étape de destruction étant effectuée lorsque l'étape de détection détecte une anomalie d'exécution.

Ce mode de réalisation est particulièrement avantageux pour protéger la carte contre des attaques par perturbation du microcircuit, qui de façon connue ne permettent pas facilement de cibler une instruction particulière, du moins à la première tentative.

Dans cette variante de réalisation, l'étape de destruction peut par exemple détruire le système d'exploitation de la carte à microcircuit ; la carte à microcircuit devient alors inutilisable après détection d'une attaque.

Dans une variante préférée de réalisation, la première signature est générée automatiquement, par exemple par un outil de génie logiciel.

Cette caractéristique permet avantageusement de simplifier le travail du développeur du programme informatique à sécuriser.

Selon un deuxième aspect, la présente invention vise un dispositif de traitement d'un programme informatique comportant un ensemble d'au moins une instruction et caractérisé en ce qu'il comporte :

-des moyens de calcul et de mémorisation, préalablement à ladite exécution, d'une première signature représentative de l'exécution attendue dudit ensemble d'instructions.

Ce dispositif de traitement permet la génération automatique de la première signature du programme informatique.

Il peut notamment être adapté à calculer et mémoriser une première signature constituée par un élément ou une combinaison d'éléments obtenus à partir :

-du nombre d'instructions de l'ensemble d'instructions ; ou

-du code ou de l'adresse d'une instruction critique de l'ensemble d'instructions ; ou

-d'un code détecteur d'erreur calculé à partir du code ou d'une adresse d'au moins une instruction de l'ensemble d'instructions, ce code détecteur d'erreur pouvant être un code de redondance cyclique.

Selon un autre aspect, la présente invention concerne un dispositif de sécurisation de l'exécution d'un programme informatique comportant un ensemble d'au moins une instruction. Ce dispositif de sécurisation comporte :

-un premier registre de mémorisation d'une première signature représentative de l'exécution attendue de l'ensemble d'instructions;

-des moyens de calcul et de mémorisation, au cours de l'exécution de l'ensemble d'instructions, d'une deuxième signature représentative de l'exécution de l'ensemble d'instructions, cette mémorisation étant effectuée dans un deuxième registre de mémorisation ; et

-des moyens de détection d'une anomalie d'exécution de l'ensemble d'instructions à partir des première et deuxième signatures.

L'invention vise aussi une carte à puce comportant un dispositif de sécurisation tel que décrit brièvement ci-dessus.

Les avantages du dispositif de traitement, du dispositif de sécurisation et de la carte à puce selon l'invention étant identique à ceux du procédé de sécurisation décrit précédemment, ils ne seront pas rappelés ici.

L'invention sera mieux comprise et d'autres avantages apparaîtront plus clairement à la lumière de la description qui va suivre d'un procédé et d'un dispositif de sécurisation ainsi que d'un dispositif de traitement d'un programme informatique conformes à son principe, cette description étant

donnée uniquement à titre d'exemple et faite en référence aux dessins annexés dans lesquels :

- la figure 1 représente les principales étapes d'un procédé de sécurisation de l'exécution d'un programme informatique conforme à l'invention dans un mode particulier de réalisation ;
- la figure 2 représente de façon schématique un dispositif de traitement d'un programme informatique conformément à la présente invention dans un mode particulier de réalisation ; et
- la figure 3 représente un dispositif de sécurisation d'un programme informatique conforme à l'invention dans un mode préféré de réalisation.

La description est par ailleurs accompagnée des annexes A à D qui comportent quatre exemples de programmes informatiques sécurisés conformément à l'invention en langage assembleur.

L'**annexe A** comporte le code en assembleur d'un programme informatique A. Ce code assembleur comporte 13 instructions notées A1 à A13.

Au cours de l'instruction A1, on mémorise dans le registre opcode_count la valeur 101.

Cette instruction est suivie par une instruction A2 au cours de laquelle on initialise le bit opcode_start avec la valeur 1, ce qui déclenche la décrémentation automatique du registre opcode_count à chaque nouvelle instruction exécutée.

L'instruction A2 est suivie par trois instructions A3, A4, A5 au cours desquelles on mémorise respectivement la valeur message1 dans le registre r0, la valeur message2 dans le registre r1 et la valeur 16 dans le registre r2.

L'instruction A5 est ensuite suivie par une boucle « boucle_round_i » constituée par les instructions A6 à A11.

Plus précisément, au cours de l'instruction A6, on mémorise dans un registre a la donnée pointée par l'adresse contenue dans le registre r1. L'instruction A6 est suivie par une instruction A7 au cours de laquelle on réalise

l'opération OU exclusif logique « XOR » entre le contenu du registre a et la donnée pointée par l'adresse contenue dans le registre r0.

L'instruction A7 est suivie par une instruction A8 au cours de laquelle on remplace la donnée pointée par l'adresse contenue dans le registre r0 par le contenu du registre a.

L'instruction A8 est suivie par deux instructions A9 et A10 au cours desquelles on incrémente respectivement d'une unité le contenu des registres r0 et r1.

L'instruction A10 est suivie par une instruction A11 au cours de laquelle on décrémente d'une unité le contenu du registre r2 et on teste si le contenu du registre r2 ainsi décrémenté est nul. Lorsque ce n'est pas le cas, l'instruction A11 est suivie par l'instruction A6 décrite précédemment.

Le contenu du registre r2 ayant été initialisé avec la valeur 16 à l'instruction A5, la boucle « boucle_round_i » est exécutée seize fois.

En revanche, lorsque le contenu du registre r2 est nul, l'instruction A11 est suivie par l'instruction A12 au cours de laquelle on modifie le contenu du bit opcode_start avec la valeur 0, ce qui interrompt la décrémentation automatique du compteur opcode_count.

L'instruction A12 est suivie par l'instruction A13 au cours de laquelle on décrémente le contenu du registre opcode_count, puis on teste le contenu de ce registre.

Lorsque le contenu de ce registre est non nul, le programme appelle une routine de traitement des anomalies d'exécution « error_on_opcode_count ».

Le fait de positionner à 1 la valeur du bit opcode_start au cours de l'instruction A2 permet, automatiquement, de déclencher le compteur opcode_count qui se décrémente automatiquement d'une unité à l'exécution de chaque instruction.

En pratique, le processeur exécutant le programme informatique comporte des moyens adaptés à décrémenter ce compteur à chaque chargement FETCH du code d'une opération.

Ce compteur est arrêté par la mise à zéro du bit opcode_start à l'instruction A12.

L'exécution de ce programme informatique A est sécurisée conformément à l'invention. En effet :

Au cours de la génération de ce programme, non décrite ici, une première signature de valeur 101 a été calculée et mémorisée, cette première signature étant représentative du nombre d'instructions que le programme informatique A doit normalement exécuter lorsque le bit opcode_start est positionné à 1, c'est-à-dire entre les instructions A3 et A12.

Si le programme s'exécute normalement, les instructions s'exécutant entre les instructions A3 et A12 sont :

- les instructions A3, A4 et A5 qui s'exécutent une fois ; et
- les instructions A6, A7, A8, A9, A10, A11 qui s'exécutent seize fois ; et
- l'instruction A12 qui s'exécute une fois.

Le nombre d'instructions total est donc (3 x 1) +(16 x 6)+(1 x 1) = 100.

A l'issue de l'instruction A12, le registre opcode_start contient donc si le programme est exécuté normalement, la valeur 1, (101 – 100).

Au cours de l'instruction A13, on décrémente le registre opcode_count. Le registre opcode_count doit alors normalement contenir la valeur 0. Si tel n'est pas le cas, cela signifie que toutes les instructions du programme n'ont pas été exécutées, ou que des instructions non prévues ont été exécutées.

Conformément à la présente invention, la valeur 101 mémorisée dans le registre opcode_count à l'instruction A1 est une première signature représentative de l'exécution de l'ensemble des instructions A2 à A13 calculée, par exemple, par le programmeur ou de façon automatique au moment de la génération de cet ensemble d'instructions.

De la même façon, toujours conformément à la présente invention, au cours de l'exécution du programme, le registre opcode_count est décrémenté au cours de l'exécution de chaque instruction. Il mémorise ainsi

une deuxième signature égale au nombre d'instructions prévues et n'ayant pas été exécutées, cette deuxième signature étant calculée par différence à partir de la première signature 101.

L'instruction A13 constitue quant à elle une étape de détection d'une anomalie d'exécution au cours de laquelle on vérifie si le contenu du registre opcode_count est nul, après décrémentation de ce registre. Si la valeur de ce registre est non nul, cela signifie soit qu' au moins une des instructions prévues n'a pas été exécutée, soit qu'une instruction non prévue a été exécutée, ce qui dans tous les cas constitue une anomalie.

Dans un mode préféré de réalisation, on déclenche une interruption logicielle lorsque le compteur devient strictement négatif. Ceci permet avantageusement la détection d'une attaque qui aurait pour effet de ne pas exécuter l'instruction A13 de détection et l'instruction A12 d'arrêt du compteur.

Dans ce mode de réalisation, les instructions A12 d'arrêt du compteur et A13 de détection peuvent être combinées en une seule instruction ajoutée au jeu d'instructions du processeur. Il est alors encore plus difficile de mener une attaque consistant à arrêter le compteur et à sauter l'instruction de détection.

L'annexe B comporte le code assembleur d'un autre programme informatique B.

Dans ce mode de réalisation, le processeur qui exécute le programme B comporte un registre SFR_OPH qui contient le code d'opération de la dernière instruction exécutée.

Au cours d'une première instruction B1, on réalise l'opération XRL (OU exclusif logique) entre le contenu d'un registre A et celui d'un registre R1 du processeur.

L'instruction B1 est suivie par l'instruction B2 au cours de laquelle le programme se branche au label « résultat_faux » si le contenu des registres A et R1 diffèrent.

Lorsque les contenus des registres A et R1 sont égaux, l'instruction B2 est suivie par une instruction B3 au cours de laquelle on

mémorise dans le registre R5 le contenu du registre SFR_OPH, soit le code d'opération de la dernière instruction exécutée (B2) avant l'instruction en cours d'exécution (B3).

Ainsi, à l'issue de l'instruction B3, le registre R5 contient le code de l'instruction JNZ, soit 70h.

L'instruction B3 est ensuite suivie par trois instructions B4 à B6 au cours desquelles, respectivement, on mémorise dans le registre A la valeur du registre R7, on réalise un OU logique entre le contenu du registre A et la valeur hexadécimale 55h et on mémorise dans le registre R7 le nouveau contenu de la valeur A.

L'instruction B6 est suivie par une instruction B7 au cours de laquelle on mémorise dans le registre A, le contenu du registre R5. Conformément à ce qui a été décrit précédemment, le registre A contient le code 70h de l'opération JNZ si l'instruction B2 a été exécutée précédemment à l'instruction B3.

L'instruction B7 est suivie par une instruction B8 au cours de laquelle on réalise un OU exclusif logique entre le contenu du registre A et la valeur 70h du code de l'instruction JNZ.

Ainsi, si le test prévu à l'instruction B2 a été effectivement exécuté, le résultat du OU exclusif logique effectué à l'opération B8 est égal à 0.

Si tel n'est pas le cas, l'instruction B9 consécutive à l'instruction B8 branche le programme B sur une routine « os_kill_card » de destruction du système d'exploitation de la carte à microcircuit.

Ainsi, le programme permet de vérifier, conformément au procédé de sécurisation selon l'invention que l'instruction critique B2 a effectivement été exécutée.

Pour cela, on a, au cours de la génération du programme informatique, calculé et mémorisé dans un registre de la mémoire ROM contenant le code exécutable du programme informatique B, ce registre étant associé à l'instruction B8, une première signature représentative de l'exécution de l'instruction B2, à savoir le code 70H de l'instruction JNZ.

Au cours de l'exécution de l'instruction B3, on a mémorisé une deuxième signature en chargeant dans le registre R5 le code de l'instruction précédent l'instruction B3, soit normalement le code 70h si aucune attaque n'a été effectuée.

L'étape B9 constitue quant à elle une étape de détection d'une anomalie d'exécution, une anomalie étant détectée si la première signature et la deuxième signature diffèrent.

Le procédé précédemment décrit en référence à l'annexe B permet ainsi de détecter que l'exécution de l'instruction B2 a effectivement eu lieu.

L'annexe C comporte le code assembleur d'un autre programme informatique C.

De façon connue, chaque instruction du programme C comporte en première colonne l'adresse de cette instruction.

Par exemple, l'adresse de l'instruction C1 est 0x8500.

Dans ce mode de réalisation, le processeur qui exécute le programme C comporte un registre SFR_JMP qui mémorise l'adresse de la dernière instruction ayant exécuté un saut (JMP, JNZ, CJNE, ...).

Au cours de la première instruction C1, on réalise le OU exclusif logique XRL entre le contenu du registre A et celui du registre R1.

L'instruction C1 est suivie par une instruction C2 au cours de laquelle on effectue un saut au label « continue » si le résultat de l'instruction C1 précédente diffère de la valeur 0.

Ainsi, le programme informatique C se branche au label « continue » si les valeurs des registres A et R1 diffèrent.

Au cours de cette même instruction C2, et conformément à la présente invention, le processeur mémorise automatiquement l'adresse 0x8501 de l'instruction de saut JNZ (C2) dans le registre prédéterminé SFR_JMP, constitué d'une partie haute SFR_JMPH et d'une partie basse SFR_JMPL.

Lorsque le programme se branche au label « continue » il exécute tout d'abord deux instructions C4 et C5 respectivement au cours desquelles, il

mémorise dans le contenu du registre A la valeur 25 et il ajoute au contenu de ce registre A la valeur 34.

Il exécute ensuite, consécutivement à l'instruction C5, une instruction C6 qui termine l'exécution de ce programme informatique C.

En revanche, lorsque le résultat de l'opération OU exclusif logique exécutée à l'instruction C1 est nul, c'est à dire lorsque les contenus des registres A et R1 sont égaux, l'instruction de saut conditionnel C2 est sans effet. Cette instruction C2 est alors suivie par une instruction C3 de saut au label « code_secret ».

Conformément à la présente invention, au cours de l'exécution de l'instruction C3, le processeur mémorise automatiquement dans le registre SFR_JMP la valeur de l'adresse de l'instruction C3 à savoir 0x8503.

Lorsque le programme informatique C se branche au label « code_secret », il exécute tout d'abord deux instructions C7 et C8 au cours desquelles on mémorise respectivement dans les registres A et R5, le contenu de la partie haute SFR_JMPH et de la partie basse SFR_JMPL du contenu du registre SFR_JMP.

Normalement, si l'instruction C7 a été exécutée consécutivement à l'instruction C3, les contenus des registres A et R5 doivent respectivement contenir les valeurs 85 et 03, correspondant aux parties hautes et basses de l'adresse 0x8503.

L'instruction C8 est ensuite suivie par deux instructions C9 et C10 au cours desquelles on vérifie si le contenu des registres A et R5 est respectivement égal à 85H et 03H.

Si tel n'est pas le cas, le programme informatique exécute une routine « os_kill_card » de destruction du système d'exploitation de ce programme informatique.

Lorsque les contenus des registres A et R5 sont respectivement égaux à 85H et 03H, l'instruction C10 est suivie par une instruction C11 au cours de laquelle on mémorise dans le registre A la valeur PIN.

L'instruction C11 est suivie par une instruction C12 de fin d'exécution du programme C.

Ainsi, par exemple durant la génération du programme informatique, conformément à la présente invention, on a calculé et mémorisé les premières signatures constituées par les valeurs 85H et 03H constituant l'adresse de l'instruction C3.

Puis, au cours de l'exécution des instructions C7 et C8 on a mémorisé une deuxième signature en chargeant dans les registres A et R5 le contenu du registre SFR_JMP, ce registre devant contenir la valeur 0x8503 si le programme C s' est exécuté normalement.

Conformément à la présente invention, les instructions C9 et C10 constituent les étapes de détection d'une anomalie d'exécution, ces étapes consistant à comparer le contenu des registres A et R5, à savoir la deuxième signature, avec la première signature mémorisée au moment de la génération du programme à savoir l'adresse de l'instruction C3.

Le procédé de sécurisation décrit ici permet de vérifier que lorsque le programme se branche à l'instruction C7 il a précédemment exécuté l'instruction C3, ce qui signifie que l'opération du OU exclusif logique réalisé à l'instruction C1 a donné une valeur nulle, et donc que les registres A et R1 contiennent une valeur identique.

Le programme de sécurisation permet donc de vérifier que lorsque l'on entre dans la sous-partie du programme repéré par le label « code_secret », on exécute ces instructions de façon licite et non pas après une attaque, auquel cas la valeur du registre SFR_JMP serait différente de la valeur 0x8503, correspondant à l'adresse de l'instruction C3.

Dans un autre mode de réalisation non décrit ici, le programme informatique C comporte d'autres instructions similaires aux instructions C9 et C10, ces instructions comportant, en lieu et place des valeurs 85H et 03H, les parties basses et hautes des adresses d'autres instructions de saut, ce qui permet de sécuriser, selon le principe décrit ci-dessus un programme comportant plusieurs sauts d'instructions.

L'**annexe D** comporte le code assembleur d'un autre programme informatique D. Pour faciliter la description, les instructions des lignes D2 à D12

comportent, en première colonne, le code hexadécimal de l'instruction correspondante.

Par exemple, le code de l'instruction "mov r0,# message1" est la valeur 78 H.

Au cours de la première instruction D1, on initialise le bit chk_opcode_ start avec la valeur 1.

L'instruction D1 est suivie par trois instructions D2, D3, D4 au cours desquelles on mémorise respectivement la valeur message1 dans le registre r0, la valeur message2 dans le registre r1, et la valeur 16 dans le registre r2.

L'instruction D4 est ensuite suivie par une boucle « boucle_round_i » constituée par les instructions D5 à D10.

Plus précisément, au cours de l'instruction D5, on mémorise dans un registre a la donnée pointée par l'adresse contenue dans le registre r1.

L'instruction D5 est suivie par une instruction D6 au cours de laquelle on réalise le OU exclusif logique « XRL » entre le contenu du registre a et la donnée pointée par l'adresse contenue dans le registre r0.

L'instruction D6 est suivie par une instruction D7 au cours de laquelle on remplace la donnée pointée par l'adresse contenue dans le registre r0 par le contenu du registre a.

L'instruction D7 est suivie par deux instructions D8 et D9 au cours desquelles on incrémente respectivement d'une unité le contenu des registres r0 et r1.

L'instruction D9 est suivie par une instruction D10 au cours de laquelle on décrémente d'une unité le contenu du registre r2 et on teste si le contenu du registre r2 ainsi décrémenté est nul. Lorsque ce n'est pas le cas, l'instruction D10 est suivie par l'instruction D5 décrite précédemment.

Le contenu du registre r2 ayant été initialisé avec la valeur 16 à l'instruction D4, la boucle "boucle_round_i" est exécutée 16 fois.

En revanche lorsque le contenu du registre r2 est nul, l'instruction D10 est suivie par l'instruction D11 au cours de laquelle on mémorise dans le registre a le contenu d'un registre "checksum".

Conformément à l'invention, ce registre "checksum" est un code détecteur d'erreur calculé à partir du code des instructions D2 à D11.

En effet, dans le mode de réalisation décrit ici, après l'initialisation (au cours de l'instruction D1) du bit chk_opcode_start avec la valeur 1, le processeur réalise le OU exclusif logique entre le code des instructions suivantes et mémorise le résultat de cette opération dans le registre checksum.

Ainsi, à l'issue de l'exécution de l'instruction D2, le contenu du registre checksum est égal à 78h.

De même, à l'issue de l'exécution de l'instruction D3, le contenu du registre checksum est égal au OU exclusif logique entre les valeurs 78h et 79h soit 01.

Ainsi, à l'issue de la boucle constituée des opérations D5 à D10, le registre checksum contient une signature représentative des instructions qui ont été exécutées après l'exécution de l'instruction D1.

En l'espèce, si les instructions D2, D3, D4 ont été exécutées une fois et si les instructions D5 à D10 ont été exécutées 16 fois, le contenu du registre checksum à l'issue de la dernière exécution de l'instruction D10 est égale à 7Bh.

L'instruction D11 est suivie par une instruction D12 au cours de laquelle on compare le contenu du registre a avec la valeur hexadécimale 0Fh.

Cette valeur 0Fh constitue, selon les termes de la présente invention, une première signature calculée et mémorisée dans un registre de la mémoire comportant le programme D, cette première signature étant représentative de l'exécution attendue des instructions D2 à D11.

Au cours de l'instruction D12, si on constate que le contenu du registre a est différent de la valeur 0Fh, le programme D appelle une routine de traitement des anomalies d'exécution "kill_card".

Cette instruction D12 constitue ainsi une étape de détection d'une anomalie d'exécution des instructions D2 à D11 à partir de la première signature 0Fh et de la deuxième signature mémorisée dans le registre checksum.

Dans le mode préféré de réalisation décrit ici, la routine de traitement "kill_card" est une routine rendant l'utilisation de la carte à microcircuit impossible.

La **figure 1** représente les principales étapes E10 à E80 d'un procédé de sécurisation conforme à la présente invention dans un mode préféré de réalisation.

Dans l'exemple décrit ici, le procédé de sécurisation permet de sécuriser l'exécution d'un programme informatique EXE dont le code source SOURCE est écrit en langage assembleur.

Il s'agit par exemple d'un code source semblable à ceux décrits précédemment en référence aux annexes A à D.

Le procédé de sécurisation selon l'invention comporte deux séries d'étapes, à savoir une première série composée des étapes E10 à E30 de génération du code exécutable du programme informatique à partir du code source SOURCE et une seconde série d'étapes constituée par les étapes E40 à E80 d'exécution du programme informatique EXE.

Au cours de la première étape E10 du procédé de sécurisation selon l'invention, on lit le code source SOURCE du programme à sécuriser.

Cette étape de lecture d'un code source est connue de l'homme du métier. Elle peut être mise en œuvre par un assembleur ou compilateur.

L'étape E10 de lecture du code source est suivie par une étape E20 au cours de laquelle on génère le code exécutable EXE du programme informatique.

Cette étape est connue de l'homme du métier. Elle peut notamment être mise en œuvre par un compilateur.

L'étape E20 de génération du code exécutable EXE est suivie par une étape E30 de calcul et de mémorisation d'une première signature SIG1 représentative de l'exécution attendue d'un ensemble d'instructions du programme informatique à sécuriser.

Dans le mode de réalisation décrit ici, cette première signature SIG1 est mémorisée dans un registre REG0 de la mémoire morte ROM comportant le programme informatique exécutable EXE.

Dans une autre variante de réalisation, cette étape E30 de calcul et de mémorisation d'une première signature SIG1 s'effectue au cours de la lecture du code source E10.

Dans une autre variante de réalisation, cette étape E30 de calcul et de mémorisation d'une première signature SIG1 s'effectue au cours de l'étape de génération du code exécutable E20. C'est particulièrement le cas, lorsque cette première signature est obtenue à partir de l'adresse d'une instruction critique du programme informatique.

Cette première étape E30 de calcul et de mémorisation peut être effectuée de différentes manières.

Dans une première variante de réalisation, la première signature SIG1 est le nombre d'instructions dont l'exécution est à sécuriser. Une telle variante de réalisation a précédemment été décrite en référence à l'annexe A.

Plus précisément, lorsque le langage de programmation du programme informatique est le langage assembleur, cette étape consiste à compter le nombre des codes d'opérations des instructions de l'ensemble d'instructions à sécuriser.

Dans la variante de réalisation dans laquelle le langage de programmation est un langage de haut niveau tel le langage C, cette étape consiste tout d'abord à générer un programme assembleur à partir du code source de haut niveau, puis, à compter le nombre d'instructions en assembleur ainsi générées.

Les outils de génie logiciel, et en particulier l'interface graphique de ces outils, permettant à un programmeur d'introduire dans le programme informatique les instructions nécessaires à la mise en œuvre du procédé de sécurisation conforme à l'invention seront décrits ultérieurement en référence à la figure 2.

Dans l'exemple décrit précédemment en référence à l'annexe A, la première signature est le nombre 101 mémorisé dans un registre REG0 de la mémoire comportant le code exécutable du programme A.

Dans une autre variante de réalisation, telle que décrite précédemment en référence à l'annexe B, la première signature est le code d'une instruction critique de l'ensemble d'instructions.

Ainsi, dans le programme B décrit précédemment, la première signature est le code 70h de l'instruction JNZ. Cette première signature est mémorisée dans un registre REG0 de la mémoire du programme exécutable B.

Dans un autre mode de réalisation, tel que décrit en référence à l'annexe C, la première signature est l'adresse d'une instruction critique de l'ensemble d'instructions, cette adresse étant obtenue pendant ou après la génération du code exécutable de l'ensemble d'instructions.

Dans l'exemple décrit précédemment en référence à l'annexe C, cette première signature est la valeur 0x8503 correspondant à l'adresse de l'instruction C3, cette adresse étant mémorisée dans des registres de la mémoire comportant le code exécutable du programme C, ces registres étant associés aux instructions C9 et C10.

Dans un autre mode préféré de réalisation, tel que décrit en référence à l'annexe D, la première signature est un code détecteur d'erreurs calculé à partir du code d'au moins une instruction de l'ensemble d'instructions à sécuriser.

Cette signature peut par exemple être une combinaison logique des codes d'instructions à sécuriser.

Cette première signature peut également être, dans un mode préféré de cette variante de réalisation, un code de redondance cyclique obtenu à partir de ces codes d'instructions, ou par combinaison logique (XOR) de ces codes d'instruction.

L'étape E30 est suivie par une étape E40 au cours de laquelle on débute l'exécution du programme informatique EXE.

Dans le mode de réalisation décrit ici, ce programme informatique EXE est mémorisé dans une mémoire morte ROM dont un registre REG0 comporte la première signature SIG1 calculée à l'étape E20.

L'étape E40 de début d'exécution du programme informatique est suivie par une étape E45 au cours de laquelle on copie le contenu SIG1 du

registre REG0 de la mémoire morte ROM dans un registre REG1 de la mémoire vive RAM utilisée par le programme informatique EXE.

L'étape de copie E45 est suivie par une étape E50 au cours de laquelle on calcule et on mémorise une deuxième signature SIG2 dans un registre REG2 de la mémoire vive RAM, cette deuxième signature SIG2 étant représentative de l'exécution proprement dite de l'ensemble d'instructions à mémoriser.

Cette deuxième signature est calculée au cours de l'exécution de l'ensemble d'instructions.

Il s'agit par exemple du nombre d'instructions ayant été exécutées à un instant donné ou du nombre d'instructions restant à exécuter tel que décrit précédemment en référence à l'annexe A.

Il peut également s'agir, tel que décrit en référence à l'annexe B, du code d'une instruction venant d'être exécutée, ou, comme décrit en référence à l'annexe C, d'une adresse de cette instruction.

Dans le mode de réalisation dans lequel la première signature SIG1 est un code détecteur d'erreur calculé à partir du code d'au moins une instruction de l'ensemble d'instructions, l'étape E50 de calcul et de mémorisation de la deuxième signature SIG2 est effectuée en calculant le code détecteur d'erreur au fur et à mesure de l'exécution des instructions considérées.

Il peut notamment s'agir, comme précisé précédemment en référence à l'annexe D, d'un code de redondance cyclique ou d'une combinaison logique telle que XOR.

Quoiqu'il en soit, l'étape E50 de calcul et mémorisation d'une deuxième signature SIG2 permet de mémoriser la deuxième signature dans un registre REG2 de la mémoire vive RAM d'exécution du programme informatique EXE.

Préférentiellement, le contenu de ce registre REG2 est conservé pendant l'exécution d'au moins une deuxième instruction consécutive à l'ensemble d'instructions à sécuriser.

L'étape E50 de calcul et de mémorisation d'une deuxième signature SIG2 est suivie par un test E60 de détection d'une anomalie d'exécution de l'ensemble d'instructions à partir des première et deuxième signatures, respectivement SIG1 et SIG2.

Conformément à l'invention, un nombre quelconque d'instructions peuvent être exécutées entre l'étape E50 et le test E60.

Ce test E60 de détection d'une anomalie consiste à vérifier si la valeur de la deuxième signature SIG2 est en accord avec la valeur de la première signature SIG1 calculée et mémorisée au cours de la génération du programme informatique, ce qui signifie que les instructions à sécuriser ont été exécutées conformément à ce qui était prévu.

Lorsque la deuxième signature SIG2 n'est pas en accord avec la première signature SIG1, cela signifie que le programme informatique EXE n'a pas été exécuté tel que prévu, par exemple suite à une attaque DFA.

Le test E60 de détection d'une anomalie d'exécution peut notamment consister à comparer les signatures SIG1 et SIG2 lorsque ces signatures sont :

- le nombre d'instructions prévues SIG1 et le nombre d'instructions réellement exécutées SIG2 ;
- le code SIG1 d'une instruction critique prévue et le code SIG2 d'une instruction réellement exécutée ;
- l'adresse SIG1 d'une instruction critique prévue et l'adresse SIG2 d'une instruction réellement exécutée ; et
- un code détecteur d'erreurs CRC1 par exemple un code de redondance cyclique calculé à partir des codes d'opération d'un certain nombre d'instructions et le code détecteur d'erreurs CRC2, par exemple un code de redondance cyclique calculé à partir d'un ensemble d'instructions réellement exécutées.

Dans le mode préféré de réalisation décrit précédemment à l'annexe A, cette étape E60 de détection d'une erreur d'exécution consiste à comparer le nombre d'instructions restant à exécuter avec la valeur zéro.

Quoiqu'il en soit, lorsqu'une anomalie d'exécution est détectée au cours du test E60 de la détection, ce test est suivi par une étape E70 de traitement de cette anomalie.

Cette étape E70 peut notamment consister en une étape de destruction d'une partie du système informatique dans lequel le programme informatique s'exécute, par exemple la destruction du système d'exploitation, ce qui rend la carte à microcircuit inutilisable en cas de détection d'une attaque mal intentionnée.

En variante, l'étape E70 de traitement est mise en œuvre uniquement lorsque plusieurs anomalies ont été détectées au cours des différentes utilisations de la carte. On utilise alors pour cela un registre dans lequel sera accumulé le nombre d'anomalies, éventuellement en fonction de leur nature.

En variante, l'étape de traitement E70 consiste à réinitialiser le système informatique dans lequel le programme informatique s'exécute, par exemple par une commande reset.

Dans une autre variante de réalisation, l'étape E70 de traitement d'une anomalie peut consister en l'envoi d'un simple message d'information destiné à l'utilisateur du dispositif.

Cette variante de réalisation est particulièrement intéressante lorsque le procédé de sécurisation selon l'invention est utilisé pour fiabiliser le déroulement d'un programme informatique (satellite, contrôle d'un implant cardiaque).

Cette étape E70 termine le procédé de sécurisation conforme à la présente invention décrit ici.

Par ailleurs, lorsque aucune anomalie n'est détectée au cours de l'étape E60 de détection d'une anomalie d'exécution, le test E60 est suivi par une étape E80 où d'autres instructions, non sécurisées, sont éventuellement exécutées.

A l'issue de l'étape E80, le procédé de sécurisation se termine.

La **figure 2** représente de façon schématique un dispositif de traitement d'un programme informatique conforme à la présente invention dans un mode préféré de réalisation.

Dans un mode préféré de réalisation, le dispositif 10 de traitement comporte des moyens 11 d'édition du programme informatique SOURCE.

Ces moyens 11 d'édition comportent en particulier des moyens de sélection, par exemple un clavier, une souris et un traitement de texte, permettant de sélectionner, dans le programme informatique SOURCE, les instructions à sécuriser.

Ces moyens 11 d'édition permettent également d'introduire dans le programme informatique SOURCE, de façon automatique ou manuelle, des instructions particulières pour le calcul et la mémorisation de la deuxième signature SIG2, cette deuxième signature SIG2 étant calculée au cours de l'exécution des instructions devant être sécurisées.

Ces moyens 11 d'édition permettent aussi d'introduire dans le programme informatique SOURCE une instruction permettant de mettre en œuvre l'étape de détection d'une anomalie.

Le dispositif 10 de traitement d'un programme informatique comporte des moyens 12 de calcul et de mémorisation, au cours de la génération des instructions, d'une première signature SIG1 représentative de l'exécution attendue de l'ensemble d'instructions sélectionnées par les moyens d'édition 11.

Ces moyens 12 de calcul et de mémorisation sont, de façon connue, adaptés à lire le programme informatique SOURCE. Ils sont en particulier adaptés à lire un programme informatique tels que ceux donnés aux annexes A à D.

Dans un mode préféré de réalisation, le dispositif 10 de traitement comporte des moyens 14 de génération d'un code exécutable à partir du programme informatique source.

Ces moyens 14 de génération d'un code exécutable sont en particulier adaptés à obtenir et à mémoriser les adresses des instructions

critiques à sécuriser dans le code exécutable à partir du programme informatique SOURCE.

Les moyens 12 de calcul et de mémorisation sont, adaptés, dans un mode de réalisation préféré, à compter le nombre d'instructions sélectionnées et le nombre d'arguments de ces instructions par les moyens 11 d'édition, et à mémoriser cette valeur dans un registre.

Cette valeur constitue une première signature SIG1 conformément à la présente invention.

Les moyens 12 de calcul et de mémorisation sont également adaptés à mémoriser le code d'opération d'une instruction critique sélectionnée par les moyens 11 d'édition.

Ils sont également adaptés à mémoriser l'adresse d'une instruction sélectionnée par les moyens 11 d'édition, cette adresse ayant été obtenue par les moyens 14 de génération d'un code exécutable pendant ou après la génération du code exécutable.

Dans une variante de réalisation, les moyens 12 de calcul et de mémorisation sont adaptés à calculer un code détecteur d'erreurs, par exemple un code de redondance cyclique ou une combinaison logique (XOR) à partir du code des instructions sélectionnées par les moyens 11 d'édition.

Dans une variante de réalisation, le registre REG1 de mémorisation de la première signature SIG1 peut également être sélectionné par les moyens 11 d'édition.

Quoiqu'il en soit, le dispositif de traitement génère un programme informatique EXE susceptible d'être exécuté ultérieurement et associé à un registre REG0 mémorisant une signature SIG1 représentative de l'exécution attendue des instructions sélectionnées par les moyens d'édition 11.

Nous allons maintenant décrire en référence à la **figure 3** une carte à microcircuit C comportant un dispositif de sécurisation 100 conforme à la présente invention dans un mode préféré de réalisation.

Le dispositif 100 de sécurisation de la carte à microcircuit C selon l'invention décrit ici comporte un registre REG1 de mémorisation d'une première signature SIG1 représentative de l'exécution attendue d'un ensemble

d'instructions à sécuriser, cette signature ayant été calculée et mémorisée par un dispositif de traitement 10 du type de ceux décrits précédemment en référence à la figure 2.

Le dispositif 100 de sécurisation comporte en outre des moyens 22 de calcul et de mémorisation au cours de l'exécution de l'ensemble d'instructions d'une deuxième signature SIG2 représentative de l'exécution proprement dite des instructions, cette mémorisation étant effectuée dans un deuxième registre REG2 de mémorisation.

Dans un mode préféré de réalisation, les moyens 22 de calcul et de mémorisation sont adaptés à compter le nombre d'instructions de l'ensemble d'instructions ayant été exécutées, ou celles n'ayant pas encore été exécutées, par comparaison avec la signature SIG1.

Préférentiellement, les moyens 22 de calcul et de mémorisation comportent des moyens de déclenchement d'une interruption du programme informatique EXE lorsque la deuxième signature SIG2 est inférieure à un seuil prédéterminé.

Ces moyens de déclenchement d'une interruption sont connus de l'homme du métier et ne seront pas décrits ici.

Dans l'exemple décrit précédemment en référence à l'annexe A, les instructions de l'ensemble d'instructions à sécuriser sont celles s'exécutant alors que le bit opcode_start est positionné à 1, c'est à dire entre l'exécution des instructions A2 et A12 insérées dans le programme informatique SOURCE par les moyens d'édition 11 du dispositif 10 de génération décrit en référence à la figure 2.

Ainsi, lorsque le processeur du dispositif de sécurisation exécute ces instructions, les moyens 22 de calcul et de mémorisation décrémentent le compteur opcode_count contenu dans le deuxième registre REG2.

En pratique, cette opération est réalisée par le processeur au cours de l'instruction FETCH.

Dans un deuxième mode de réalisation, les moyens 22 de calcul et de mémorisation mémorisent dans le registre REG2 le code ou l'adresse de la dernière instruction exécutée.

Ceci peut se faire tel que décrit précédemment, respectivement en référence aux instructions D7, C7 et C8 des annexes B et C.

Dans un autre mode de réalisation, les moyens 22 de calcul et de mémorisation calculent un code détecteur d'erreur à partir du code ou d'une adresse d'au moins une instruction de l'ensemble d'instructions.

En pratique, le calcul du code de détecteur d'erreur est réalisé par le processeur qui exécute le programme informatique à sécuriser au moment du chargement en mémoire des instructions sélectionnées en utilisant les moyens 11 de sélection du dispositif de traitement selon l'invention.

Dans le mode préféré de réalisation de cette variante, le code détecteur d'erreur est un code de redondance cyclique CRC2 ou une combinaison logique (XOR) calculés à partir des codes ou des adresses de ces instructions.

Le dispositif 100 de sécurisation selon l'invention comporte également des moyens 24 de détection d'une anomalie d'exécution de l'ensemble d'instructions à sécuriser.

Ces moyens 24 de détection sont par exemple constitués d'un comparateur des registres REG1 et REG2.

Ce mode préféré de réalisation permet de détecter en particulier une différence entre les signatures SIG1 générées et mémorisées par le dispositif 10 de génération selon l'invention et la signature SIG2 calculée et mémorisée dans le registre REG2 par les moyens 22 de calcul et de mémorisation du dispositif de sécurisation 100.

Ils permettent en particulier de détecter une erreur dans les modes de réalisation dans lesquels les signatures sont constituées :

- par le nombre d'instructions de l'ensemble d'instructions ;
- par l'adresse d'une instruction critique de cet ensemble d'instructions ;
- ou du code d'une instruction critique de cet ensemble.

Dans un autre mode de réalisation préféré, les moyens 24 de détection d'une anomalie sont constitués par un comparateur adapté à

comparer le contenu du registre REG2 de mémorisation de la deuxième signature avec la valeur 0.

Ce mode préféré de réalisation est facile à mettre en œuvre ; il permet de détecter que toutes les instructions à sécuriser ont été exécutées, tel que précédemment décrit en référence à l'annexe A.

Dans un mode préféré de réalisation, les moyens 22 de calcul et de mémorisation sont adaptés à conserver la deuxième signature dans le deuxième registre REG2 pendant l'exécution d'au moins une instruction consécutive à l'ensemble d'instructions à sécuriser.

Dans cette variante préférée, le deuxième registre REG2 peut par exemple être un registre dédié d'une mémoire volatile RAM du dispositif de sécurisation 100, ce registre de mémoire volatile étant écrasé par les moyens 24 de détection d'une anomalie après comparaison des première et deuxième signature.

Dans le mode préféré de réalisation décrit ici, les première SIG1 et deuxième signatures SIG2 sont mémorisées dans le même registre REG1 de la mémoire vive RAM au cours de l'exécution du programme informatique EXE.

ANNEXE A

```
A1      mov     opcode_count, #101
A2      setb    opcode_start

A3      mov     r0, #message1
A4      mov     r1, #message2
A5      mov     r2, 16
        boucle_round_i :
A6      mov     a,@r1
A7      xrl     a, @r0
A8      mov     @r0, a
A9      inc     r0
A10     inc     r1
A11     djnz    r2, boucle_round_i

A12     clrb    opcode_start
A13     djnz    opcode_count, error_on_opcode_count
```

ANNEXE B

```
B1      XRL     A, R1
B2      JNZ     résultat_faux

    résultat_bon :
B3      MOV     R5, SFR_OPH

B4      MOV     A, R7
B5      ORL     A, #55h
B6      MOV     R7, A
B7      MOV     A,R5
B8      XRL     A, #70h

B9      JNZ     os_kill_card
B10     RET

    résultat_faux :

B11     MOV     A, R6
B12     RET
```

ANNEXE C

```
C1      0x8500      XRL     A, R1
C2      0x8501      JNZ     continue

C3      0x8503      SJMP    code_secret

                continue :
C4      0x8505      MOV     A, #25
C5      0x8507      ADD     A, #34
C6      0x8509      RET


                code_secret :
C7      0x850A      MOV     A, SFR_JMPH

C8      0x850C      MOV     R5, SFR_JMPL
C9      0x850E      CJNE    A, #85h, os_kill_card
C10     0x8511      CJNE    R5, #03h, os_kill_card

C11     0x8514      MOV     A, PIN
C12     0x8516      RET
```

ANNEXE D

```
D1                  setb    chk_opcode_start

D2      78          mov     r0, #message1
D3      79          mov     r1, #message2
D4      7A          mov     r2, 16
boucle_round_i :
D5      E7          mov     a,@r1
D6      66          xrl     a,@r0
D7      F6          mov     @r0,a
D8      08          inc     r0
D9      09          inc     r1
D10     DA          djnz    r2, boucle_round_i

D11     74          mov     a, checksum
D12     B4          cjne    a, #0F,kill_card
```

REVENDICATIONS

1. Procédé de sécurisation de l'exécution d'un programme informatique (EXE) comportant un ensemble d'au moins une instruction, caractérisé en ce qu'il comporte :

- une première étape (E30) de calcul et de mémorisation, préalable à l'exécution dudit programme informatique, d'une première signature (SIG1) représentative de l'exécution attendue dudit ensemble d'instructions ;
- une deuxième étape (E50) de calcul et de mémorisation, au cours de l'exécution dudit ensemble d'instructions, d'une deuxième signature (SIG2) représentative de l'exécution dudit ensemble d'instructions ; et
- une étape (E60) de détection d'une anomalie d'exécution dudit ensemble d'instructions à partir desdites première (SIG1) et deuxième (SIG2) signatures.

2. Procédé de sécurisation selon la revendication 1, caractérisé en ce que ladite première étape (E30) de calcul et de mémorisation a lieu au cours de la génération des instructions (A1, A13) dudit programme informatique.

3. Procédé de sécurisation selon la revendication 1 ou 2, caractérisé en ce que ladite deuxième signature (SIG2) mémorisée au cours de ladite deuxième étape de calcul et de mémorisation (E50) est conservée en mémoire pendant l'exécution d'au moins une deuxième instruction consécutive audit ensemble d'instructions.

4. Procédé de sécurisation selon l'une quelconque des revendications 1 à 3, caractérisé en ce que :

-la première signature (SIG1) est obtenue à partir du nombre d'instructions dudit ensemble d'instructions ;

-la deuxième signature (SIG2) est obtenue à partir du nombre d'instructions dudit ensemble d'instructions ayant été exécutées ; et en ce que

-l'étape de détection (E60) détecte une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

5. Procédé de sécurisation selon l'une quelconque des revendications 1 à 3, caractérisé en ce que :

-la première signature (SIG1) est obtenue à partir du nombre d'instructions dudit ensemble d'instructions ;

-la deuxième signature (SIG2) est obtenue à partir du nombre d'instructions dudit ensemble d'instructions n'ayant pas été exécutées, cette deuxième signature (SIG2) étant calculée à partir de ladite première signature (SIG1) ; et en ce que

-l'étape de détection (E60) détecte une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la deuxième signature (SIG2) est non nulle.

6. Procédé de sécurisation selon la revendication 5, caractérisé en ce que l'on déclenche une interruption dudit programme informatique lorsque ladite deuxième signature (SIG2) est inférieure à un seuil prédéterminé.

7. Procédé de sécurisation selon la revendication 5 ou la revendication 6, caractérisé en ce que la première (SIG1) et la deuxième (SIG2) signatures sont conservées en mémoire pendant l'exécution dudit programme dans le même registre (REG1).

8. Procédé de sécurisation selon l'une quelconque des revendications 1 à 3, caractérisé en ce que :

-la première signature (SIG1) est obtenue à partir du code d'une d'instruction critique dudit ensemble d'instructions ;

-la deuxième signature est obtenue à partir du code de ladite instruction critique, ce code étant mémorisé simultanément ou consécutivement à l'exécution de ladite instruction critique ; et en ce que

-l'étape de détection (E60) détecte une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

9. Procédé de sécurisation selon l'une quelconque des revendications 1 à 3, caractérisé en ce que :

-la première signature (SIG1) est obtenue à partir de l'adresse d'une d'instruction critique dudit ensemble d'instructions, ladite adresse étant obtenue pendant ou après la génération du code exécutable de l'ensemble d'instruction ;

-la deuxième signature (SIG2) est obtenue à partir de l'adresse de ladite instruction critique, cette adresse étant mémorisée (E30) simultanément ou consécutivement à l'exécution (E30) de ladite instruction critique ;

-l'étape de détection (E60) détecte une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

10 Procédé selon la revendication 8 ou 9 caractérisé en ce que ladite instruction critique est une instruction de saut (JMP, JNZ, CJNE, JZ).

11. Procédé de sécurisation selon l'une quelconque des revendications 1 à 3, caractérisé en ce que :

-la première (SIG1) et la deuxième (SIG2) signatures sont des codes (CRC1, CRC2) détecteurs d'erreur calculés à partir du code ou d'une adresse d'au moins une instruction dudit ensemble d'instructions ; et en ce que

-l'étape de détection (E60) détecte une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

12. Procédé de sécurisation selon la revendication 11, caractérisé en ce que lesdits codes détecteurs d'erreur (CRC1, CRC2) sont des codes de redondance cyclique.

13 Procédé de sécurisation selon la revendication 11, caractérisé en ce que lesdits codes détecteurs d'erreur sont obtenus par combinaison logique (XOR) à partir du code ou d'une adresse d'au moins une instruction dudit ensemble d'instructions.

14. Procédé de sécurisation selon l'une quelconque des revendications 1 à 3, caractérisé en ce que :

-la première signature (SIG1) et la deuxième signature (SIG2) sont obtenues, respectivement au cours de la génération et de l'exécution desdites instructions, à partir d'au moins deux éléments choisis parmi :

-le nombre d'instructions dudit ensemble d'instructions ;

-le code d'au moins une instruction dudit ensemble d'instructions ;

-l'adresse d'au moins une instruction dudit ensemble d'instructions ; et

- un code détecteur d'erreur calculé à partir du code ou d'une adresse d'au moins une instruction critique dudit ensemble d'instructions, ladite adresse étant obtenue pendant ou après la génération du code exécutable de l'ensemble d'instruction ; et en ce que

-l'étape de détection (E60) détecte une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

15. Procédé de sécurisation selon l'une quelconque des revendications 1 à 14, caractérisé en ce qu'il comporte une étape (E70) de destruction d'au moins une partie du système sur lequel s'exécute ledit programme informatique, cette étape de destruction étant effectuée lorsque ladite étape de détection détecte ladite anomalie d'exécution.

16. Procédé de sécurisation selon l'une quelconque des revendications 1 à 15, caractérisé en ce que ladite première signature (SIG1) est générée automatiquement (E30).

17. Dispositif de traitement d'un programme informatique comportant un ensemble d'au moins une instruction, caractérisé en ce qu'il comporte

-des moyens (12) de calcul et de mémorisation, préalablement à ladite exécution, d'une première signature (SIG1) représentative de l'exécution attendue dudit ensemble d'instructions.

18. Dispositif de traitement selon la revendication 17, caractérisé en ce que lesdits moyens (12) de calcul et de mémorisation de ladite première signature (SIG1) sont adaptés à calculer et mémoriser une information obtenue à partir du nombre d'instructions dudit ensemble d'instructions.

19. Dispositif de traitement selon la revendication 17, caractérisé en ce que par lesdits moyens (12) de calcul et de mémorisation de ladite première signature (SIG1) sont adaptés à obtenir et mémoriser une information obtenue à partir du code d'une d'instruction critique dudit ensemble d'instructions.

20. Dispositif de traitement selon l'une des revendications 17 à 19, caractérisé en ce qu'il comporte en outre des moyens (14) de génération d'un code exécutable à partir dudit programme informatique (SOURCE).

21. Dispositif de traitement selon la revendication 20, caractérisé en ce lesdits moyens de calcul et de mémorisation de ladite première signature (SIG1) sont adaptés à obtenir et mémoriser une information obtenue à partir de l'adresse d'une instruction critique dudit ensemble d'instructions, ladite information étant obtenue par lesdits moyens (14) de génération d'un code exécutable.

22. Dispositif de traitement selon la revendication 19 ou 21, caractérisé en ce que ladite instruction critique est une instruction de saut (JMP, JNZ, CJNE, JZ).

23. Dispositif de traitement selon la revendication 17, caractérisé en ce que lesdits moyens (12) de calcul et de mémorisation de ladite première signature (SIG1) sont adaptés à calculer et mémoriser une information obtenue à partir d'un code détecteur d'erreur (CRC1) calculé à partir du code ou d'une adresse d'au moins une instruction dudit ensemble d'instructions.

24. Dispositif de traitement selon la revendication 23, caractérisé en ce que ledit code détecteur d'erreur (CRC1) est un code de redondance cyclique.

25. Dispositif de traitement selon la revendication 23, caractérisé en ce que ledit code détecteur d'erreur est obtenu par combinaison logique (XOR) à partir du code ou d'une adresse d'au moins une instruction dudit ensemble d'instructions.

26. Dispositif de sécurisation de l'exécution d'un programme informatique comportant un ensemble d'au moins une instruction, caractérisé en ce qu'il comporte :

-un premier registre (REG1) de mémorisation d'une première signature (SIG1) représentative de l'exécution attendue dudit ensemble d'instructions ;

-des moyens (22) de calcul et de mémorisation, lors de l'exécution dudit ensemble d'instructions, d'une deuxième signature (SIG2) représentative de l'exécution dudit ensemble d'instructions, ladite mémorisation étant effectuée dans un deuxième registre (REG2) de mémorisation ;

-des moyens (24) de détection d'une anomalie d'exécution dudit ensemble d'instructions à partir desdites première (SIG1) et deuxième (SIG2) signatures.

27. Dispositif de sécurisation selon la revendication 26, caractérisé en ce que les moyens de calcul et de mémorisation sont adaptés à conserver ladite deuxième signature (SIG2) dans le deuxième registre (REG2) lors de l'exécution d'au moins une deuxième instruction consécutive audit ensemble d'instructions.

28. Dispositif de sécurisation selon la revendication 26 ou la revendication 27, caractérisé en ce que, ladite première signature (SIG1) étant obtenue à partir du nombre d'instructions dudit ensemble d'instructions, ladite deuxième signature (SIG2) calculée et mémorisée par lesdits moyens (22) de calcul et de mémorisation est obtenue à partir du nombre d'instructions dudit ensemble d'instructions ayant été exécutées, et en ce que lesdits moyens (24) de détection détectent une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, les première (SIG1) et deuxième signatures (SIG2) sont différentes.

29. Dispositif de sécurisation selon la revendication 26 ou la revendication 27, caractérisé en ce que, ladite première signature (SIG1) étant obtenue à partir du nombre d'instructions dudit ensemble d'instructions, ladite deuxième signature (SIG2) calculée et mémorisée par lesdits moyens (22) de calcul et de mémorisation est obtenue à partir du nombre d'instructions dudit ensemble d'instructions n'ayant pas été exécutées, cette deuxième signature (SIG2) étant calculée à partir de ladite première signature (SIG1), et en ce que lesdits moyens (24) de détection détectent une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la deuxième signature (SIG2) est non nulle.

30. Dispositif de sécurisation selon la revendication 29, caractérisé en ce qu'il comporte en outre des moyens de déclenchement d'une interruption dudit programme informatique lorsque ladite deuxième signature (SIG2) est inférieure à un seuil prédéterminé.

31. Dispositif de sécurisation selon la revendication 29 ou la revendication 30, caractérisé en ce que les première (SIG1) et deuxième (SIG2) signatures sont mémorisées pendant l'exécution dudit programme dans ledit premier registre (REG1).

32. Dispositif de sécurisation selon la revendication 26 ou la revendication 27, caractérisé en ce que, ladite première signature (SIG1) étant obtenue à partir du code d'une d'instruction critique dudit ensemble d'instructions, ladite deuxième signature (SIG2) calculée et mémorisée par lesdits moyens (22) de calcul et de mémorisation est obtenue à partir du code de ladite instruction critique, ce code étant mémorisé simultanément ou consécutivement à l'exécution de ladite instruction critique, et en ce que lesdits moyens de détection (24) détectent une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

33. Dispositif de sécurisation selon la revendication 26 ou la revendication 27, caractérisé en ce que, ladite première signature (SIG1) étant obtenue à partir de l'adresse d'une d'instruction critique dudit ensemble d'instructions, ladite deuxième signature (SIG2) calculée et mémorisée par lesdits moyens (22) de calcul et de mémorisation étant obtenue à partir de l'adresse de ladite instruction critique, cette adresse étant mémorisée simultanément ou consécutivement à l'exécution de ladite instruction critique, et en ce que lesdits moyens de détection détectent une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, les première et deuxième signatures sont différentes.

34. Dispositif de sécurisation selon la revendication 32 ou la revendication 33, caractérisé en ce que ladite instruction critique est une instruction de saut (JMP, JNZ, CJNE, JZ).

35. Dispositif de sécurisation selon la revendication 26 ou la revendication 27, caractérisé en ce que, lesdites première (SIG1) et deuxième (SIG2) signatures étant des codes détecteurs d'erreur (CRC1, CRC2) calculés à partir du code d'au moins une instruction dudit ensemble d'instructions, lesdits moyens de détection (24) détectent une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble d'instructions, la première (SIG1) et la deuxième (SIG2) signatures sont différentes.

36. Dispositif de sécurisation selon la revendication 35, caractérisé en ce que lesdits codes détecteurs d'erreur (CRC1, CRC2) sont des codes de redondance cyclique.

37. Dispositif de sécurisation selon la revendication 35 caractérisé en ce que lesdits codes détecteurs d'erreur sont obtenus par combinaison logique (XOR) à partir du code ou d'une adresse d'au moins une instruction dudit ensemble d'instruction.

38. Dispositif de sécurisation selon la revendication 26 ou la revendication 27 caractérisé en ce que caractérisé en ce que ladite première signature (SIG1) étant obtenue à partir d'au moins deux éléments choisis parmi :

- le nombre d'instructions dudit ensemble d'instructions ;
- le code d'au moins une instruction dudit ensemble d'instructions ;
- l'adresse d'au moins une instruction dudit ensemble d'instructions ; et
- et un code détecteur d'erreur calculé à partir du code ou de l'adresse d'au moins une instruction dudit ensemble d'instructions,

ladite deuxième signature (SIG2) calculée et mémorisée par lesdits moyens (22) de calcul et de mémorisation est obtenue, de façon similaire, à partir desdits au moins deux éléments au cours de l'exécution desdites instructions et en ce que lesdits moyens (24) de détection détectent une anomalie d'exécution lorsque, à l'issue de l'exécution dudit ensemble

d'instructions, la première (SIG1) et la deuxième signatures (SIG2) sont différentes.

39. Dispositif de sécurisation selon l'une quelconque des revendications 26 à 38 caractérisé en ce qu'il comporte en outre des moyens de destruction d'au moins une partie dudit programme informatique.

40. Carte à microcircuit caractérisé en ce qu'il comporte un dispositif de sécurisation (100) selon l'une quelconque des revendications 26 à 39.

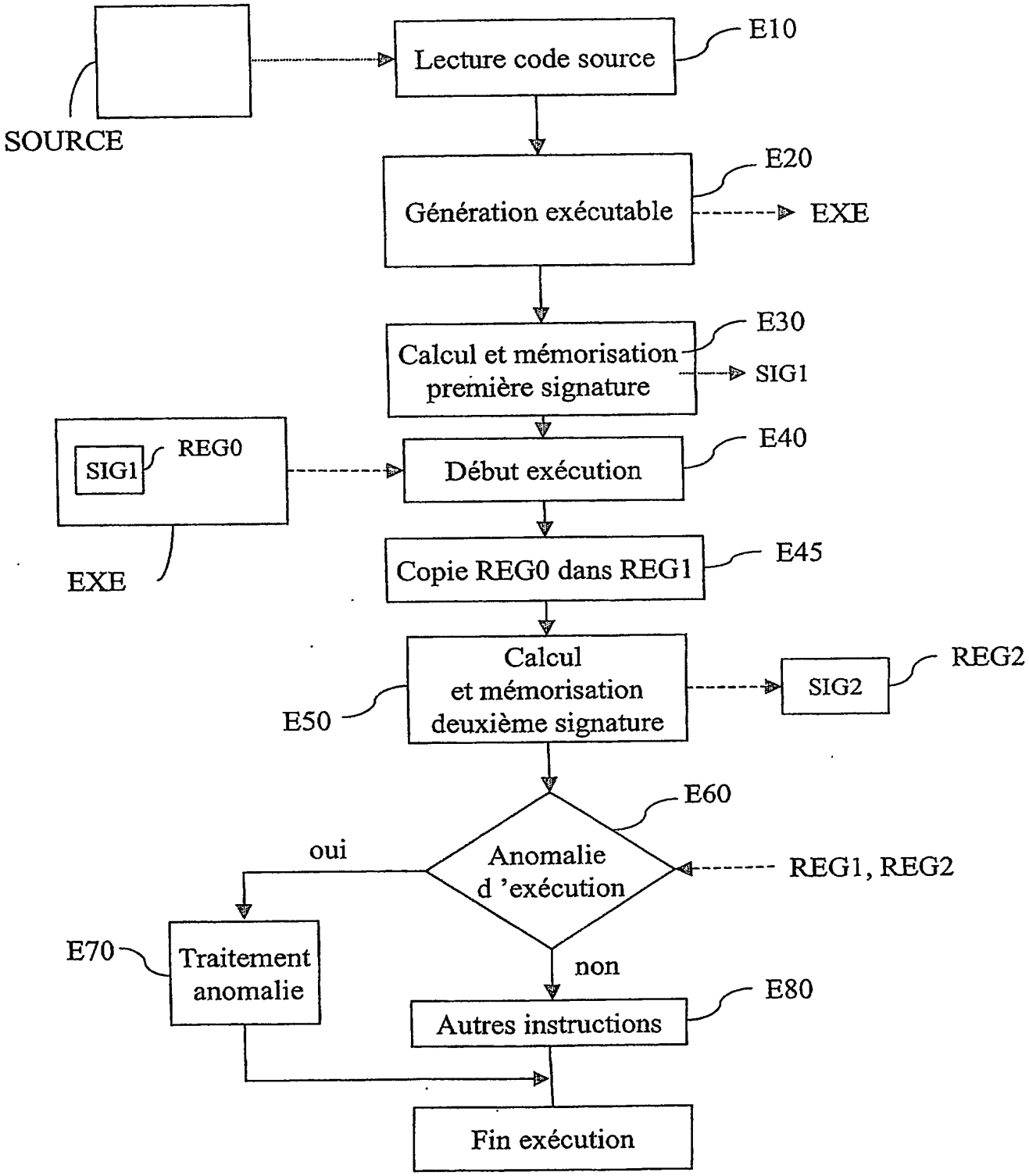
SOURCE
Lecture code source
E10
Génération exécutable
E20
EXE
Calcul et mémorisation première signature
E30
SIG1
SIG1
REG0
EXE
Début exécution
E40
Copie REG0 dans REG1
E45
Calcul et mémorisation deuxième signature
E50
SIG2
REG2
Anomalie d 'exécution
E60
REG1, REG2
oui
non
Traitement anomalie
E70
Autres instructions
E80
Fin exécution

FIGURE 1

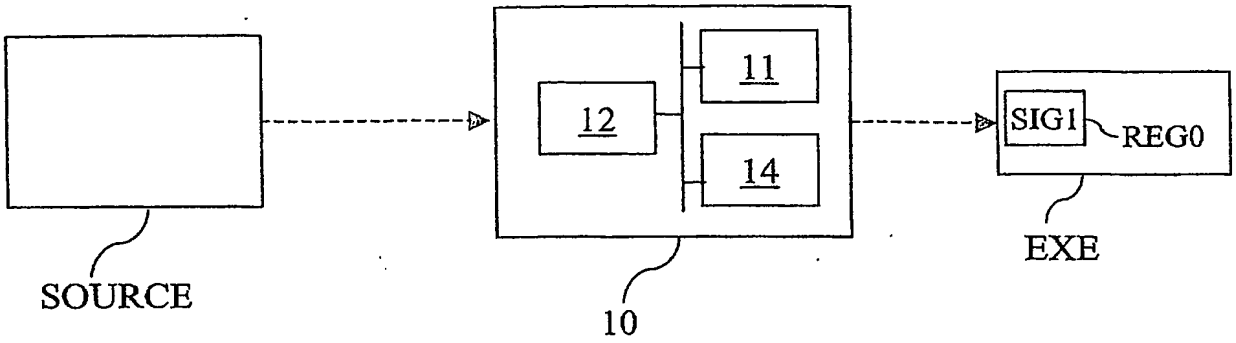

FIGURE 2

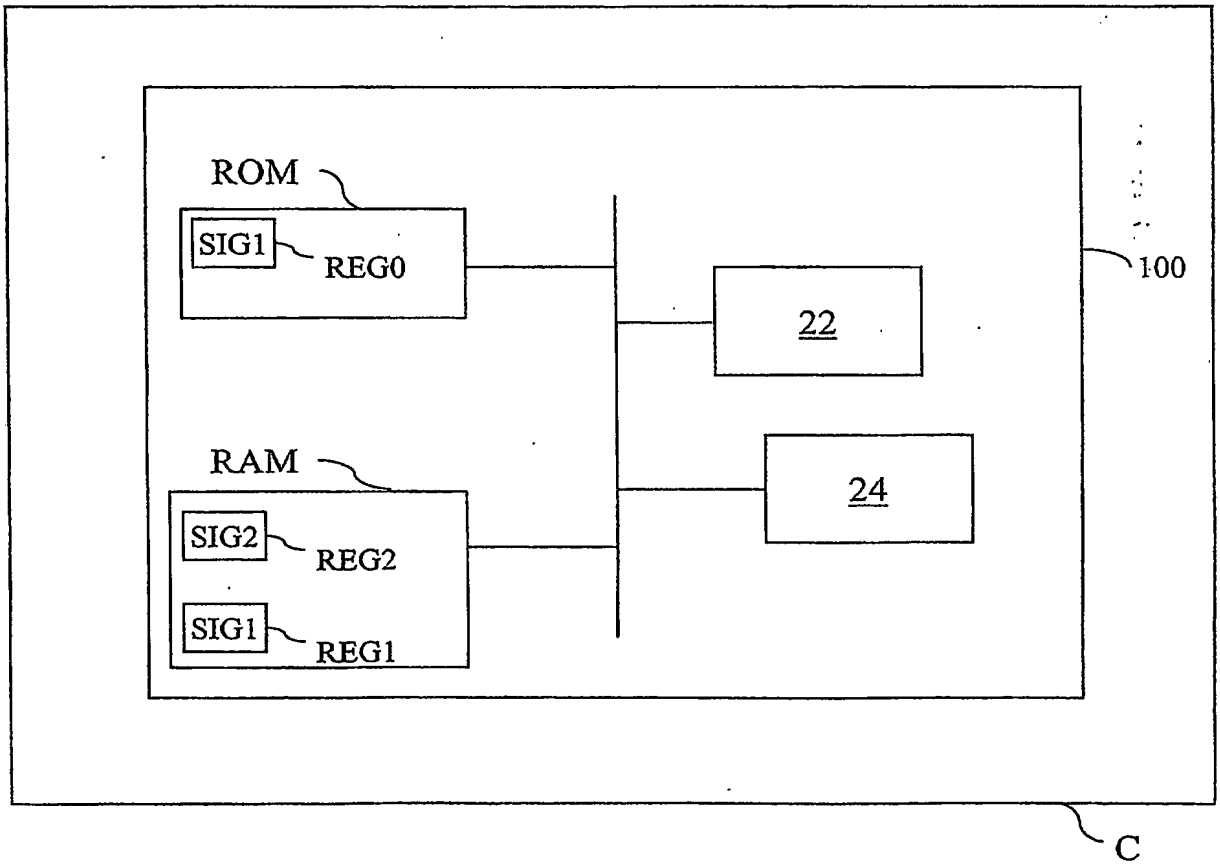

FIGURE 3

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 33 (1) 53 04 53 04 Télécopie : 33 (1) 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

N° 11235*03

DÉSIGNATION D'INVENTEUR(S) Page N° 1/1

INV

(À fournir dans le cas où les demandeurs et les inventeurs ne sont pas les mêmes personnes)

Cet imprimé est à remplir lisiblement à l'encre noire

Vos références pour ce dossier (facultatif)	BIFIT4681/FR
N° D'ENREGISTREMENT NATIONAL	0216376

TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Procédé et dispositif de sécurisation de l'exécution d'un programme informatique.

LE(S) DEMANDEUR(S) :

OBERTHUR CARD SYSTEMS SA

DESIGNE(NT) EN TANT QU'INVENTEUR(S) :

1 Nom		FISCHER
Prénoms		Jean-Bernard
Adresse	Rue	38, rue Carnot,
	Code postal et ville	9 4 2 7 0 Le KREMLIN BICETRE, France.
Société d'appartenance (facultatif)		
2 Nom		DISCHAMP
Prénoms		Paul
Adresse	Rue	26, rue Saint Lambert,
	Code postal et ville	7 5 0 1 5 PARIS, France.
Société d'appartenance (facultatif)		
3 Nom		
Prénoms		
Adresse	Rue	
	Code postal et ville	
Société d'appartenance (facultatif)		

S'il y a plus de trois inventeurs, utilisez plusieurs formulaires. Indiquez en haut à droite le N° de la page suivi du nombre de pages.

DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom et qualité du signataire)

Le 20 Décembre 2002
Bruno QUANTIN N°92.1206
CABINET BONNET-THIRION

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire.
Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☑ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.